Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2693 - Laboratorio de Algoritmos y Estructuras III
Trimestre Septiembre-Diciembre 2017

# Laboratorio de la semana 6

## 1. Introducción

En este laboratorio deberán resolver el problema que se presenta a continuación[1], usando el algoritmo **DFS con Backtracking**.

## 2. Actividad a realizar: Sumando Árboles

El lenguaje LISP fue uno de los primeros lenguajes de programación de alto nivel que aún se utiliza. Las listas que son la principal estructura de datos de LISP pueden ser fácilmente adaptadas para representar otras estructuras de datos como árboles.

Este laboratorio ejercita el determinar si un árbol binario, representado como expresiones-S de LISP posee cierta propiedad.

Dado un árbol binario de enteros, deberán escribir un programa que determine si existe un camino desde la raíz hasta una hoja cuyos nodos sumen un entero especificado. Por ejemplo, en el árbol de la Figura 1 hay exactamente cuatro caminos raíz-a-hoja. Las sumas de estos caminos son 27, 22, 26 y 18.

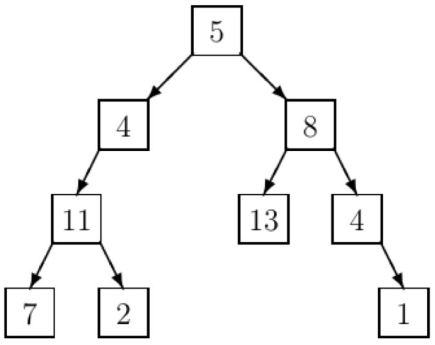


Figura 1: Ejemplo de árbol binario de enteros

Los árboles binarios se representan en el archivo de entrada como expresiones-S de LISP que tienen la siguiente forma:

```
empty tree ::= ()tree ::= empty tree | (integer tree tree)
```

El árbol del ejemplo de la Figura 1 se representa con la expresión:

```
(5 (4 (11 (7 () ()) (2 () ()) ) ()) (8 (13 () ()) (4 () (1 () ()) ) ) )
```

[1]Este problema está tomado del ACM International Collegiate Programming Contest (https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=48)

Noten que en esta formulación todas las hojas del árbol tienen la forma:

```
(integer () () )
```

Como un árbol vacío no tiene caminos raíz-a-hoja, cualquier consulta sobre si existe una ruta cuya suma sea un entero específico debe responderse negativamente.

## 3. Entrada

La entrada consiste en una secuencia de casos de prueba en la forma de pares entero/árbol. Cada caso de prueba consiste en un entero, seguido por uno a mas espacios en blanco, seguidos por un árbol binario en la forma de una expresión-S descritas en la sección anterior.

Todas las expresiones-S de árboles serán válidas, sin embargo estas expresiones pueden expandirse en varias líneas y contener espacios en blanco. Un archivo de entrada puede contener uno o mas casos de prueba, y finaliza con EOF (fin de archivo).

## 4. Salida

Debe haber una línea de salida por cada caso de prueba (par entero/árbol) en el archivo de entrada. Para cada par $< E, A >$ (E representa el entero, A representa el árbol) la salida será el string "si" en el caso que si hay un camino raíz-a-hoja en A cuya suma es E y "no" en caso contrario.

## 5. Ejemplo de Entrada

```
22 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))
20 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))
10 (3   (2 (4 () () ) (8 () () ) )
        (1 (6 () () ) (4 () () ) ) )
5 ()
```

## 6. Ejemplo de Salida

```
yes
no
yes
no
```

## 7. Condiciones de entrega

Debe entregar el día 26 de Octubre de 2017 antes de las 4:30 pm, un archivo comprimido que debe ser llamado **LabSemX-NumEquipo.tar.gz**, con todos los códigos Java y el archivo Makefile. Donde NumEquipo del archivo comprimido es el número asignado al equipo.
El nombre del programa que resuelve este problema debe ser SumaArboles.java y debe recibir como parámetros el nombre de los archivos de entrada y salida en ese orden.

Ivette C. Martínez / martinez@usb.ve / 25 de Octubre de 2017